USING MATHEMATICA IN TEACHING ROMBERG INTEGRATION

Ali YAZICI
Computer Engineering Department
Atýlým University, Ankara 06836 - Turkey
e-mail: aliyazici@atilim.edu.tr

Tanýl ERGENÇ
Department of Mathematics
Middle East Technical University, Ankara 06531 - Turkey
e-mail: tanil@rorqual.cc.metu.edu.tr

Yýlmaz AKYILDIZ
Department of Mathematics
Boshporus University, Ýstanbul - Turkey
e-mail: akyildiz@boun.edu.tr

ABSTRACT

High order approximations of an integral can be obtained by taking the linear combination of lower degree approximations in a systematic way. One of these approaches for 1-d integrals is known as Romberg Integration and is based upon the composite trapezoidal rule approximations and the well-known Euler-Maclaurin expansion of the error. Because of its theoretical nature, students in a classical Numerical Analysis course usually find it difficult to follow. In order to overcome the difficulty, Mathematica software is utilized to illustrate the method, and the underlying theory. A Mathematica program and a set of experiments are designed to explain the method and it's intricacies in a stepwise manner. The program is expected to help the student to learn and apply the method to 1-d finite integrals. However, with minor modifications, it is possible to extend the method to multi-dimensional integrals.

1. Introduction

The Romberg integration is the problem of approximating the integral below using the linear combinations of well-known trapezoidal sums T_i^{1}'s in a systematic way in order to achieve higher orders in an effective manner.

$$I = \int_a^b f(x)\, dx , \qquad a,b \in \Re, \quad f \in C^k[a,b]$$

The method is based on the Euler-Maclaurin asymptotic error expansion formula and the Richardson extrapolation to the limit (Joyce 1971). Romberg, a German mathematician, (Romberg 1955) has been the first to organize the Richardson's method in a systematic way suitable for automatic calculations on the computer in 1955.

Geometrically speaking, the value of I is the area under the curve of *y=f(x)* bounded by the x-axis, and the lines x=a, and x=b. T_1^1 is the area of the trapezium and approximates the value of I as shown in Figure 1 below.

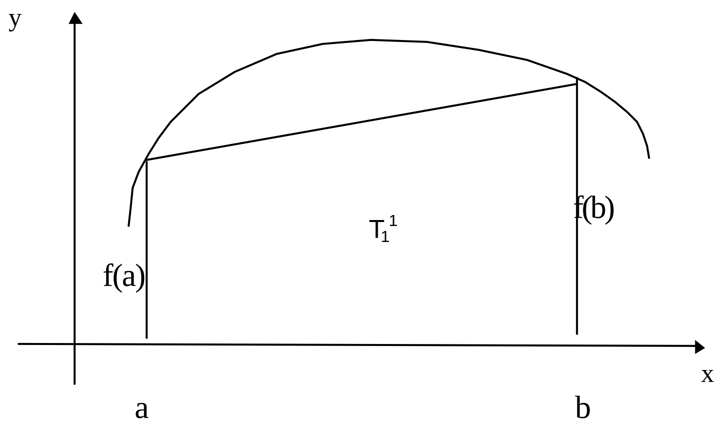

Figure 1 Basic trapezoidal computation T_1^1 **over [a,b].**

Each trapezoidal sum is defined as

$$T_i^1 = \frac{b-a}{2^i}[\, f[a] + f[b] + 2\sum_{j=1}^{2^{i-1}-1} f[x_j]]$$

for i=1,2,...,n (n ≡ maximum level of subdivision), $x_j = x_0 + jh$, j=1,2,…,i and h = $(b-a)/2^{i-1}$. Note that for the ith subdivision of the interval x_0 = a, and x_i = b. The computation starts with T_1^1 on the interval [a,b], and T_2^1, T_3^1, and so on are computed by successively halving the interval and applying the basic rule T_1^1 to each subinterval formed. In this subdivision process the Romberg sequence {1,2,4,8,16,...} is utilized. Other subdivision sequences are also possible (Yazýcý 1990).

For example, after the computation of T_1^1 as shown above, the interval of integration is bisected and the second composite approximation T_2^1 to I is formed as shown below. Obviously, as the number of subintervals increases a better, although same order of, approximations to I are obtained.

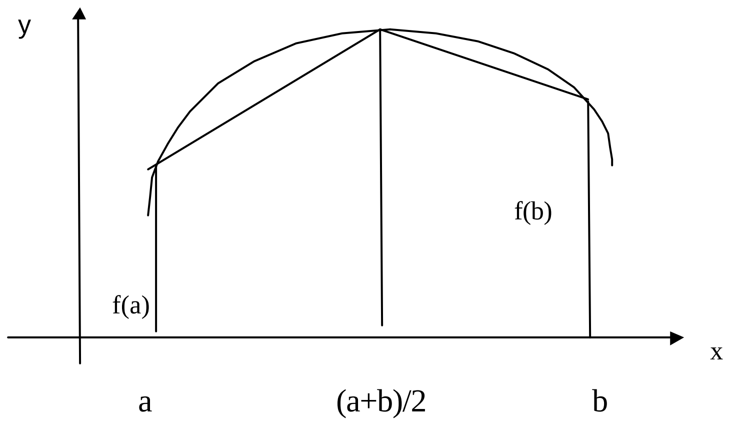

Figure 2 Composite trapezoidal sum T_2^1 over 2 subintervals.

Once the composite Trapezoidal sums are available, so-called Romberg table can be formed.

$$\begin{array}{llllll}
T_1^1 & & & & & \\
T_2^1 & T_2^2 & & & & \\
T_3^1 & T_3^2 & T_3^3 & & & \\
T_4^1 & T_4^2 & T_4^3 & T_4^4 & & \\
\vdots & \vdots & \vdots & \vdots & \ddots & \\
T_n^1 & T_n^2 & T_n^3 & T_n^4 & \cdots & T_n^n
\end{array}$$

via

$$T_i^j = \frac{4^{j-1}T_i^{j-1} - T_{i-1}^{j-1}}{4^{j-1}-1}, \quad i = 2,3,\cdots,n \quad and \quad j = 2,3,\cdots,i .$$

It is known that the entries in the second column of the table are composite Simpson's approximations to the same integral (Burden & Faires 1985). The third column entries are also composite approximations based on the Newton interpolatory formulae. The consecutive columns have no resemblance to any known method based on interpolation. The trapezoidal rule is of polynomial order one. That is, trapezoidal sums are exact whenever the integrand *f(x)* is a first-degree polynomial in x. Provided that the 1st column entries converge to I, all diagonal sequences over the table converge to I as well (Kelch 1993). Moreover, if column k is of order p then the column k+1 entries are of order p+2. This could easily be justified using the asymptotic error expansion formula:

$$I - T_i^1 = c_1h^2 + c_2h^4 + \cdots + c_kh^{2k} + O(h^{2k+2}) \quad , \quad h = \frac{b-a}{2^{i-1}} \ .$$

The c_i's are constants (based on the Bernoulli numbers) independent of h. This is an even expansion in powers of h. The linear combinations formed by the Romberg procedure causes the c_i's vanish one by one. Obviously, h approaching to zero (application of the composite rule for smaller and smaller values of h) suggests that T_j^1 converges to I. For singular integrals this expansion is not valid and it takes different forms depending on the nature of singularity (Lyness & Mc Hugh 1970).

In order to show the way c_i's vanish when the linear combination of the composite values are formed, the expansion formula above is applied with two different step sizes h_1=b-a (original interval size), and h_2=(b-a)/2, (interval size after the first bisection) to obtain

$$I - T_1^1 = c_1h_1^{\ 2} + c_2h_1^{\ 4} + \cdots + c_kh_1^{\ 2k} + O(h_1^{\ 2k+2}) \quad , \quad h_1 = b-a$$
$$I - T_2^1 = c_1h_2^{\ 2} + c_2h_2^{\ 4} + \cdots + c_kh_2^{\ 2k} + O(h_2^{\ 2k+2}) \quad , \quad h_2 = \frac{b-a}{2}$$

Multiplying both sides of the latter by 4 and subtracting from the first, and rearranging the resulting equation, one gets

$$I - \frac{4T_2^1 - T_1^1}{3} = -\frac{1}{4}c_2h_1^{\ 4} + O(h_1^{\ 6})$$

which shows that the first error term of the expansion vanishes and the linear combination of T_1^1, and T_2^1, (T_1^2 = [$4T_2^1$ - T_1^1]/3), produces a higher order approximation to I.

2. Romberg Integration with Mathematica

It is the feeling of the authors that, in learning the Romberg integration, students face some difficulties in understanding the rational behind the method. The discussion over the asymptotic error expansion and Euler-Maclaurin series and convergence makes the presentation more complicated. Working out the details of the derivations and combinations of the composite rules and the formation of the Romberg table is time consuming, if not boring. Instead, a simple symbolic program could be quite beneficiary to show all the details and derivations. Such an approach will give the student a chance to play around with the formulas and observe easily the relation between the composite sums, order of an approximation and the high orders achievable by forming the simple linear combinations.

A text-based Mathematica (Burbulla & Dodson 1992) is used to develop the program below:

```
romberg/: romberg[f_,{a_,b_},n_]:= (
```

1. Define h and initialize other variables

    h = b - a;

2. Generate array t for composite sums (to maximum level 10)

    Array[t,10];

3. Apply the basic trapezoidal rule to f

```
t[1] = h/2 (f[a] + f[b])
```

4. Create array x to hold abscissas of the points generated as a result of subdivision. The newly generated nodes (x, •, and +) utilized by t[2], t[3], and t[4] as depicted by the Romberg subdivision sequence are illustrated in Figure 3.

```
Array[x,512];
```

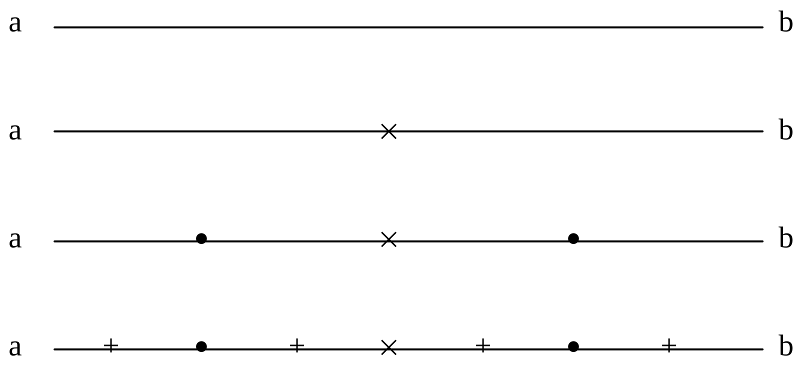

Figure 3 Nodes of subdivision at levels 1,2,3, and 4

5. Compute composite trapezoidal sums

```
For[m = 2, m <= 10, m++,
  k = m - 1;
  Do[ x[j] = a + (j -1)h/2^k , {j , 1 ,  2^k+1}];
  t[m] = h/2^m ( f[a] + f[b] +2 Sum[f[x[j]],{j , 2 , 2^k}] )
```

6. Define the Romberg Extrapolation table r (10x10 matrix) and initialize its first column to t

```
Array[ r , {10,10}]
For[m = 1 , m <= 10 , m++ , r[m,1] = t[m]]
```

7. Form the Romberg table using the first column entries

```
For[ i = 2 , i <= 10 , i++ ,
  For[ j = 2 , j <= i  , j++ ,
    r[i,j] = (4^(j-1) r[i,j-1] - r[i-1,j -1] )/(4^(j -1)-1)]]
```

Once, this program is made available to the student, the method can be investigated for a symbolic function *f* over [a,b] in an effective manner by calling the subprogram **romberg** with *f* for, say, 10 levels of subdivision as

```
f[x_] := g[x]
romberg[f,{a,b}, 10]
```

Romberg integration uses the so-called Romberg sequence $R = \{1,2,4,8,16,..., 2^k,...\}$ to subdivide the interval. Other subdivision sequences are also possible and may reduce the number of function evaluations for the same accuracy. However, Romberg sequence provides full overlapping of the nodes of integration, i.e., all the nodes at level k of subdivision are included in level k+1. This idea is incorporated in Step 5 above by replacing t[m] by a recursive definition as follows:

```
For[m = 2, m <= 10, m++,
k = m - 1;
For[j = 1, j <= 2^(k-1), j++,
Do[x[j] = a + (2j-1)h/2^k , {j, 1, 2^k+1}];
t[m] = t[m-1]/2+ h/2^k(Sum[ f[x[j]], {j, 1, 2^(k-1)}])
```

3. Experiments

A sample Mathematica session is set up to demonstrate the power of the Romberg integration for a general function *f*. The following instructions are to be carried out after setting up the definitions above.
Experiment 1: Set up the first trapezoidal approximation t[1] to I over [a,b].

In[1] := t[1]

```
               (-a+b)(g[a] + g[b])
Out[1] = -----------------------
                    2
```

Experiment 2: Set up the composite trapezoidal rule t[2] over 2 sub-intervals.

In[2] := t[2]

```
                                        -a +b
               (-a+b) (g[a] + g[b] + 2 g[a + -------])
                                          2
Out[2] = ---------------------------------------------
                              4
```

Experiment 3: Set up the composite trapezoidal rule t[3] over 4 sub-intervals.

In[3] := t[3]

```
Out[3] = ( (-a+b) (g[a] + g[b] +

            -a+b            -a+b            3(-a+b)
  2 (g[a +-------] + g[a +-------] + g[a + ----------]))))/8
             4               2               4
```

Experiment 4: Simplify the expression

In[4] := Simplify[%]

```
                                    a+b         3a+b          a+3b
               (-a+b) (g[a] + g[b] + 2 g[-----] + 2 g[------] + 2 g[-------])
                                     2           4             4
Out[4] = ----------------------------------------------------------------------
                                  8
```

Experiment 5: Set up the first Romberg value as a linear combination of t[1] and t[2] and observe that this is identical to Simpson's approximation over [a,b].

In[5] := Simplify[(4 t[2] - t[1])/3]

```
                                   a+b
             (-a+b) (g[a] + g[b] + 4 g[-----])
                                    2
Out[5] = -------------------------------------
                        6
```

Experiment 6: The Romberg table is generated and stored in the two-dimensional array r. Compare Out[5] with the value of r[2,2].

In[6] := Simplify[r[2,2]]

```
                                   a+b
             (-a+b) (g[a] + g[b] + 4 g[-----])
                                    2
Out[6] = --------------------------------------
                       6
```

Experiment 7: Display the value of r[3,2] (Simpson's rule applied to 2 sub-intervals)

In[7] := Simplify[r[3,2]]

```
                                   a+b          3a+b          a+3b
             (-a+b) (g[a] + g[b] + 2 g[-----] + 4 g[------] + 4 g[------])
                                    2             4             4
Out[7] = ------------------------------------------------------------------
                                  12
```

Experiment 8: Display the value of r[3,3] (First entry in the third column of the Romberg table). Observe that this is also an approximation based on the Newton interpolatory formula. The subsequent columns have no resemblance to any known formulae based on interpolation.

In[8] := Simplify[r[3,3]]

```
                                         a+b             3a+b              a+3b
            (-a+b) (7 g[a] + 7 g[b] + 12 g[-----] + 32 g[-------] + 32 g[-------])
                                          2               4                 4
Out[8] = -----------------------------------------------------------------------------
                                        90
```

Experiment 9: Compute the integral below numerically by displaying the value of r[6,6]. Compare the result with that of Mathematica's build-in function **Integrate**.

$$\int_0^{p} Sin[x]\; dx = 2$$

```
(* DEFINE f *)

In[9] := f[x_]:= Sin[x]

(* DEFINE END POINTS OF INTEGRATION *)

In[10] := a =0

In[11] := b = Pi

(* DISPLAY SEVERAL ROMBERG TABLE VALUES *)

In[12] := r[2,2]//N

Out[12] = 2.0944

In[13] := r[4,4]//N

Out[13] = 2.0001

(* COMPUTE ACTUAL VALUE AND DISPLAY ERROR *)

In[14] := actual = Integrate[Sin[x],{x,0,Pi}]

Out[14] = 2

In[15] := err = Abs[ actual - r[6,6] ] // N

                         -12
Out[15] = 1.32072 10
```

The values of the Romberg table, r[i,j]'s, computed by the program, are as follows:

```
0
1.5708                  2.0944
1.89612         2.00456        1.99857
1.97423         2.00027        1.99998 2.00001
1.99357         2.00002        2.              2.              2.
1.99839         2.                      2.             2.              2.              2.
```

Experiment 10: As discussed earlier, the basic Trapezoidal rule is linear and therefore integrates first-degree polynomials exactly, and each Romberg column doubles the order of approximation. To investigate this let f be x^7, over [0,1/2], and observe that r[4,j] is exact (1/2048 = 0.000488281).

```
In[16] := f[x_] := x^7

In[17] := a = 0

In[18] := b = 1/2


(* CALL ROMBERG WITH F OVER [A,B] *)
```

In[19] := romberg[f, {a,b}, 4]

In[20] := r[4,4] // N

Out[20] := 0.000488281 (exact!)

The Romberg table produced by the execution of the subprogram is as follows:

```
0.00195312
0.000991821   0.00671387
0.00626326    0.00504494    0.000493368
0.00523608    0.000489369   0.000488361   0.000488281
```

4. Justification of the Method

Romberg extrapolation method is based upon the existence of the asymptotic error expansion discussed in section 1. Mathematica can be used to illustrate how and why the method works by assuming such an expansion and symbolically deriving expressions that correspond to the entries of the Romberg table. For this purpose, let

In[21] := Array[c,4]

In[22] := e[h_] := Sum[c[i] h^(2i), {i,1,4]}

In[23] := x = (4 e[h/2] - e[h]) / 3

In[24] := y = (4 e[h/4] - e[h/2]) / 3

In[25] := Expand[Simplify[x]]

```
                  4            6             8
            - (h  c[2])    5 h c[3]    21 h c[4]
Out[25] = ----------- -  ---------  -  ------------
                  4            16            64
```

In[26] := Expand[Simplify[(16 y- x) / 15]]

```
                      6          8
            16 c[3] h      21 h  c[4])
Out[26] = ------------ + -------------
                64            1024
```

The last two results illustrate that the values in the second column of the Romberg table are O(h^4) and the third column entries are of O(h^6).

5. About the Error Term of the Trapezoidal Rule

Mathematica function **Series** can be used to verify the error term of the Trapezoidal rule given by

$$E = -\frac{h^2}{12}[f'(b) - f'(a)] + \frac{(b-a)h^4}{720} f^{(4)}(m) \quad , \quad m \in [a,b]$$

For this purpose, we investigate the error in the basic rule for the integral

$$s = \int_a^{a+h} \quad Sin[x]\ dx = -Cos[a+h] + Cos[a]$$

(* DEFINE F AND CALL ROMBERG OVER [a,a+h] *)

In[27] := f[x_] := Sin[x]

In[28] := romberg[f,{a,a+h},10]

In[29] := t[1]

```
             h (Sin[a] + Sin[a+h])
Out[29] = ---------------------------
                       2
```

In[30] := s = Integrate[Sin[x],{x,a,a+h}]

Out[30] = Cos[a] - Cos[a+h]

(* FIND THE ERROR IN t[1] *)

In[31] := e = Series[s - t[1], {h,0,3}]

```
                     3
              Sin[a] h          4
Out[31] = -----------   + O[h ]
                12
```

(* USING THE DEFINITION ABOVE FOR ERROR IN TRAP. RULE *)

In[32] := terror = -h^2/12 (Cos[a+h] - Cos[a])

```
                      2
            -(h  ( -Cos[a] + Cos[a+h]))
Out[32] = -------------------------------
                          12
```

In[33] := Series[terror,{h,0,3}]

```
                     3
              Sin[a] h          4
```

Out[34] = ------------ + O[h]
                12

The values of Out[34] and Out[31] are shown to be identical verifying the dominant term of the error formula.

6. Computational Complexity of Romberg Integration

The complexity of any numerical integration algorithm based upon interpolation is mainly depicted by the number of integrand function evaluations at the nodes of integration of the numerical rule. Romberg extrapolation described in this study is no exception. An additional cost is incurred in this case in the formation of the Romberg table, which is negligible.

The Mathematica program discussed earlier in Section 2 is a static implementation of the algorithm, i.e., for a fixed subdivision level, say, **maxlevel**, all of composite Trapezoidal sums are computed first and then the Romberg table is formed. In this case, considering the overlapping of the nodes in bisecting the interval, each level n introduces 2^n additional integrand evaluations. In higher dimensions, this may result in too many function evaluations, and hence the method may not be computationally efficient. This could be avoided by forming the rows of the Romberg table dynamically. That is, at each level, rows of the table are completed by the Romberg formula and an error test is performed to check the accuracy of the diagonal value *r[n,n]*.Whenever, the error criteria is satisfied, the algorithm terminates avoiding further unnecessary subdivisions and function evaluations. Otherwise, next composite sum is to be formed by bisecting the interval one more time.

This idea can be easily incorporated into the Mathematica code given in this work. The dynamic implementation is given below.

```
dynamic_romberg/: dromberg[f_,{a_,b_},n_, tol_]:=
(h=b-a;
 Array[t,n];
 t[1]= N[h/2(f[a]+f[b])];
 Array[x,512]; Array[r,{n,n}];
 For[m=2,m<=n,m++,
   k=m-1;
   Do[x[j]= N[a+(j-1)h/2^k],{j,1,2^k+1}];
     t[m]= N[h/2^m(f[a]+f[b]+2 Sum[f[x[j]],{j,2,2^k}])]];
     For[m=1,m<=n,m++, r[m,1]=t[m]];
       reler = 1.;
       i=2;
       While[reler >= tol && i<n,
         For[j=2,j<=i,j++,
           r[i,j]=(4^(j-1)r[i,j-1]-r[i-1,j-1])/(4^(j-1)-1)];
         reler = Abs[(r[i,i]-r[i-1,i-1])/r[i-1,i-1]];
         Print["i=",i,",",r[i,i],
               "computed relative error=", reler];
         i++
     ];
 )
```

A sample run and its output is given for the approximation of *f[x]=Sin[x]*, over [0,Pi/2].

```
f[x_]:=Sin[x]
dromberg[f,{0,Pi/2},10,0.00001]
```

```
i= 2 ,  1.002280      computed relative error =0.276142

i= 3 ,  0.999992      computed relative error =0.00228311
                                                        -6
i= 4 ,  1.000000      computed relative error =8.44274 10
```

7. Comparisons and Conclusions

In this article, Romberg extrapolation technique is illustrated using the symbolic computing facility as provided by Mathematica. Main objective of this article is to facilitate symbolic computations in order to present a highly technical method in a simplified manner. Because of the nature of the work done, numerical calculations are mostly avoided. A brief comparison of different approaches to numerical integration is outlined below.

Romberg method is built on the trapezoidal rule that is based on the linear interpolation over the two points on the interval. Higher order interpolatory rules (Newton-Cotes type formulae) can be used for high order approximations. However, the coefficients of such rules alternate in sign causing loss of accuracy. Another class of integration rules are Gaussian type that uses coefficients based on the roots of certain orthogonal polynomials over the domain of integration. Gaussian type rules provide higher degrees of accuracy compared to Newton-Cotes formulae, however, amount of work done increases dramatically because of lack of overlapping during the subdivision of the interval to obtain composite sums. Monte Carlo methods involve generating random numbers over the domain of integration, and then computing the expected value (approximation to I) by simply averaging the function values at the randomly generated points. Monte Carlo methods are suitable for N-dimensional integration for its low cost compared to the rules mentioned before. For a detailed comparison of these methods the reader is referred to, for example, (Davis & Rabinowitz 1975).

REFERENCES

-Burbulla , D.C.M. and Dodson, C.T.J., 1992, *Self-tutor for Computer Calculus Using Mathematica*, Prentice-Hall Canada Inc.

-Burden, R.L. and Faires, J.D., 1985, *Numerical Analysis*, 3rd. Ed., PWS Publishers.
-Davis,P. and Rabinowitz, P., 1975, *Methods of Numerical Integration*, Academic Press.
-Joyce, D.C., 1971, “Survey of Extrapolation Processes in Numerical Analysis”, SIAM Review, **13**, No.4, 435-490.
-Kelch, R., 1993, *Numerical Quadrature by Extrapolation with Automatic Result Verification, in Scientific Computing with Automatic result Verification*, 143-185, Academic Press, Inc.
-Lyness, J.N. and Mc Hugh, B.J.J., 1970, “On the Remainder Term in the N-Dimensional Euler-Maclaurin Expansion, Num.Math., **15**,.333-344.
-Romberg. W., 1955, “Vereinfachte Numerische Integration”, Kgl. Nordske Vid. Selsk. Forh, No.28, 30-36.
-Yazýcý, A., 1990, "On the Subdivision Sequences of Extrapolation Method of Quadrature", METU Journal of Pure and Applied Sciences, **23**, No:1, 35-51.